计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(12):135−143 [doi: 10.15888/j.cnki.csa.007712]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于网络社区发现的标签传播聚类算法①

吴清寿[1,2,3], 郭　磊[1,2], 余文森[1,2]

[1](武夷学院 数学与计算机学院, 武夷山 354300)
[2](武夷学院 认知计算与智能信息处理福建省高校重点实验室, 武夷山 354300)
[3](智慧农林福建省高校重点实验室, 福州 350002)
通讯作者: 吴清寿, E-mail: 45509111@qq.com

摘　要: 高维数据的聚类特性通常难以直接观测. 将其构建为复杂网络, 节点间的拓扑结构可以反映样本之间的关系. 对网络中的节点进行社区发现, 可实现对数据更直观的聚类. 提出一种基于网络社区发现的低随机性标签传播聚类算法. 首先, 用半径和最近邻方法将数据集构建为稀疏的全连通网络. 之后, 根据节点相似度进行节点标签预处理, 使得相似的节点具有相同的标签. 用节点的影响力值改进标签传播过程, 降低标签选择的随机性. 最后, 基于内聚度进行社区的优化合并, 提高社区的质量. 在真实数据集和人工数据集上的实验结果表明, 该算法对各种类型的数据都具有较好的适应性.

关键词: 聚类; 网络构建; 社区发现; 标签传播

引用格式: 吴清寿,郭磊,余文森.基于网络社区发现的标签传播聚类算法.计算机系统应用,2020,29(12):135–143. http://www.c-s-a.org.cn/1003-3254/7712.html

Label Propagation Clustering Algorithm Based on Network Community Detection

WU Qing-Shou[1,2,3], GUO Lei[1,2], YU Wen-Sen[1,2]

[1](School of Mathematics and Computer Science, Wuyi University, Wuyishan 354300, China)
[2](Key Laboratory of Cognitive Computing and Intelligent Information Processing of Fujian Education Institutions, Wuyi University, Wuyishan 354300, China)
[3](Key Laboratory of Smart Agriculture and Forestry of Fujian Education Institutions (Fujian Agriculture and Forestry University), Fuzhou 350002, China)

Abstract: The clustering characteristics of high-dimensional data are usually difficult to observe directly. Constructing it into a complex network, the topological structure of the network nodes can reflect the relationship between samples. Community detection of nodes in the network can achieve more intuitive clustering of data. A low randomness label propagation clustering algorithm based on network community detection is proposed. First, the data set is constructed as a sparse fully connected network using the radius and nearest neighbor methods. Then, according to the similarity of the nodes, the node labels are preprocessed to make the similar nodes have the same labels. The influence value of the nodes is used to improve the label propagation process and reduce the randomness of label selection. Finally, based on the cohesion, the community is optimized and merged to improve the quality of the community. The experimental results on

① 基金项目: 福建省自然科学基金 (2019J01835); 认知计算与智能信息处理福建省高校重点实验室开放课题基金 (KLCCIIP2018107); 智慧农林福建省高校重点实验室开放课题基金 (2019LSAF03); 福建省中青年教师教育科研项目 (JAT170608); 中央引导地方科技专项 (2018L3013); 武夷学院校科研基金 (XL1201)

Foundation item: Natural Science Foundation of Fujian Province of China (2019J01835); Open Fund of Key Laboratory of Cognitive Computing and Intelligent Information Processing of Fujian Education Institutions (Wuyi University) (KLCCIIP2018107); Open Fund of The Key Laboratory of Smart Agriculture and Forestry of Fujian Education Institutions (Fujian Agriculture and Forestry University) (2019LSAF03); Mid-Aged and Young Faculty of Education Research of Fujian Province (JAT170608); Special Fund of Central Government for Local Science and Technology Development (2018L3013); Scientific Research Fund of Wuyi University (XL1201)

收稿时间: 2020-04-24; 修改时间: 2020-05-21, 2020-06-03; 采用时间: 2020-06-15; csa 在线出版时间: 2020-11-30

real data sets and artificial data sets show that the algorithm has better adaptability to all kinds of data.

Key words: clustering; complex network; community detection; label propagation

大数据时代, 各个领域时刻都在产生大量的数据, 这些数据通常都是无标签的, 要确定每一个样本的标签通常是困难的. 机器学习算法中的无监督学习可以对无标签的数据进行学习, 以期能够揭示数据之间的联系或存在的内在规律. 聚类算法是无监督学习的代表, 可通过数据的相似属性将数据进行分组, 帮助人们增进对数据的理解, 如利用聚类技术发现具有类似功能的基因组, 检测疾病的时空分布模式等.

传统的聚类算法可大致划分为基于划分的方法, 基于层次的方法, 基于密度的方法, 基于谱图划分的方法和其他方法[1]. K-means 是分割聚类的最早也是最出名的研究, 其对初始的质心选择有较强的依赖性, 且倾向于寻找圆形集簇. K-means 只考虑了连通性, Kuwil 等[2] 提出一种重心聚类算法 (Gravity Center Clustering, GCC), 同时兼顾连通性和内聚性, 且无需提供聚类的簇数. 基于密度的方法中, DBSCAN (Density-Based Spatial Clustering of Application with Noise)[3] 可以对任意形状的数据聚类, 但确定其半径和包含的样本数量是一个难点, 且在簇间混合度较大时对样本标签误判的概率较高. 郭艳婕等[4] 提出一种改进的 GS-DBSCAN 算法, 通过计算数据的分布特性, 可自适应确定半径和半径内包含的样本数。层次聚类算法包括分裂法和凝聚法, 其中, CURE (Clustering Using REpresentative) 算法[5] 能够处理形状和尺寸差别较大的簇, 对噪音点不敏感, 但对特殊形状的类簇识别能力较差. 基于图分割方法的谱聚类 (Spectral Clustering) 算法[6] 也可以对任意形状的样本进行聚类, 且通常能够收敛于全局最优解, 但其计算的时间复杂度较高, 需要预先知道簇的数量, 构建合适的相似度矩阵是一个难点. 胡卓娅等[7] 通过构造本征间隙序列, 可确定聚类的簇数. 最新的研究中, Xie 等[8] 提出一种互为最近邻的层次聚类算法 (Reciprocal-nearest-neighbors Supported Clustering, RSC), 其假设互为最近邻居的两个样本一定会划分在同一个簇中.

对于高维数据, 要通过低维空间上的可视化观察其聚类特性是困难的, 这对传统聚类方法提出了挑战. 近年来, 基于复杂网络的机器学习方法得到了研究者的关注[9]. 将向量化的数据转换为以网络表示的数据, 其过程是无损的. 以网络表示的数据相比以向量表示的数据拥有更多的信息, 如样本之间的关系结构或者拓扑信息. 通过观测网络的拓扑结构, 更易于发现样本的聚类特性.

以 Iris 数据集为例, 从每个类别中抽取其中的前 10 个样本, 通过网络构建, 得到的结果如图 1 所示. 其中, 类别标签为 Iris-virginica 的节点中, 样本 21 和 26 的特征与类别 Iris-versicolor 非常接近, 经网络构建后, 其与类别 Iris-versicolor 中样本的联系较为紧密. 其余节点都能与同类别的节点保持稠密的连边关系.

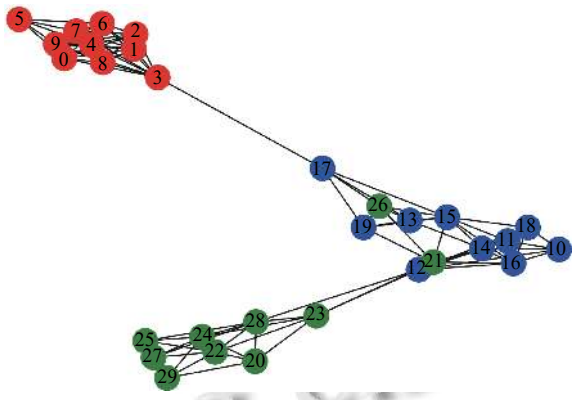


图 1 Iris 数据集部分样本的拓扑图

通过图 1 中的拓扑结构, 可以看到样本较为清晰的划分为 3 个簇, 同一簇中节点间的连边密度较大, 而簇之间的连边较为稀疏.

对向量表示的数据进行网络化后, 一种有效的聚类方法是进行社区发现[10]. 通常意义上, 社区内部的节点之间连边稠密, 而社区之间的连边稀疏.

Raghavan 等[11] 于 2007 年将一种半监督学习算法标签传播算法 (Label Propagation Algorithm, LPA)[12] 用于社区发现, 取得了良好的效果. LPA 算法具有接近线性的时间复杂度, 这对规模越来越大的社交网络研究具有重要的意义. 然而, LPA 算法为每个节点初始化一个标签, 容易造成标签传播的随机性, 并增加了传播过程的迭代次数. 将数据集进行网络化, 节点间的连边较为稀疏, 用 LPA 发现的社区数量一般远大于真实的类簇数量, 聚类准确度较差.

鉴于 LPA 具有接近的线性时间复杂度, 将 LPA 算法应用于数据聚类是一种有意义的尝试. 通过将向量表示的数据集构建为网络 (数据集中的一个样本对应网络中的一个节点), 再用改进的 LPA 算法进行社区发现, 将网络中的节点划分到不同社区, 即实现了数据集中样本的聚类. 本文提出一种低随机性的改进 LPA 算法 (Low Randomness Label Propagation Algorithm, LRLPA), 用于对网络化后的数据进行聚类. LRLPA 算法主要针对 LPA 的两个不足进行改进. (1) 对于 LPA 随机性较大的问题, 采用标签预处理和融入节点影响力的标签传播过程, 可有效降低其随机性. (2) LPA 算法在稀疏网络上可能会产生较多的社区, 本文提出一种社区内聚度的社区质量衡量指标, 对内聚度低的社区进行优化合并, 可使得划分的社区质量更高, 且社区数量更接近真实的情况. 同时, LRLPA 算法还保留了 LPA 的高效性.

1 相关工作

1.1 社区发现算法

社区发现的相关算法中, 早期, 研究者主要基于图论及矩阵论, 提出用图分割和谱分析方法进行社区划分, 如 KL 算法[13] 和谱划分算法[14,15]. 基于机器学习中聚类算法思想的延伸, Lin 等[16] 提出一种整数规划方法用于检测网络中的分层社区结构, SCAN 算法[17] 是基于密度聚类的经典社区发现算法. 基于模块度[18] 优化的社区发现算法中, BGLL[19] 在稀疏网络上有接近线性的时间复杂度, 克服了此类算法时间复杂度较大的缺点. 基于信息论的方法中, Infomap 算法[20] 利用网络上信息传播的规律来识别社区, 与 Infomap 算法类似, CDID 算法[21] 通过模拟网络中的信息交换来发现社区.

1.2 LPA 算法

LPA 算法[11] 的基本思想: 统计节点的邻居节点的社区归属情况, 某个标签对应的节点数量最多, 则选择该标签作为当前节点的标签. 其算法步骤如下.

步骤 1. 为各节点选择一个唯一的标签, 通常将各节点标签初始化为节点的编号.

步骤 2. 将所有节点随机洗牌后, 逐一更新节点标签. 根据当前节点的邻居节点标签情况, 选择对应节点数量最多的标签作为目标标签, 用以更新当前节点的标签. 如果满足条件的标签有多个, 则随机选择一个标签.

步骤 3. 重复步骤 2, 直到每一个节点的标签都不再发生变化为止.

步骤 4. 将具有相同标签的节点划分为同一个社区, 算法终止.

1.3 LeaderRank 算法

LeaderRank[22,23] 是基于 PageRank[24] 提出的用于网络中节点排序的算法. 与 PageRank 相比, 其增加了背景节点 (ground node) g. 在有向网络中, 将 g 与所有其他节点双向连接, 而在无向网络中就是将 g 与所有节点建立连边. 计算每个节点的 LR 值, 该值越大, 表示该节点越重要. 社交网络中, LR_i值可视为节点 v_i 在网络中的影响力[25].

算法初始设定 $LR_g(0)=0$, $LR_i(0)=1$, 即节点背景节点 g 的初始 LR 值为 0, 网络中的其他节点初始的重要性相同, 都为 1 单位 LR 值. 在 t 时刻, 节点 v_i 的 LR 值根据式 (1) 计算:

$$LR_i(t)=\sum_{v_j\in\Gamma_i}\frac{\delta_{ji}}{k_j^{\text{out}}}LR_j(t-1) \tag{1}$$

其中, Γ_i 表示节点 v_i 的邻居节点集合, k_j^{out} 表示节点 v_j 的出度, $LR_j(t–1)$ 表示 v_j 在 $t–1$ 时刻的 LR 值. 在有向图中, v_j 是指向 v_i 的节点, 在无向图中, v_j 是 v_i 的邻居节点, 所以 $\delta_{ij}=1$.

重复式 (1) 的计算直至 LR 值不再发生变化或变化程度小于阈值, 再根据式 (2) 修正所有节点的 LR 值.

$$LR_i=LR_i(t_c)+\frac{LR_g(t_c)}{N} \tag{2}$$

其中, t_c 是式 (1) 收敛的迭代次数, $LR_g(t_c)$ 表示节点 g 迭代至收敛状态时的 LR 值.

2 算法定义与实现

2.1 网络构建

一个包含 n 个样本的数据集表示为 $X=\{x_1, x_2, \cdots, x_n\}$, x_i 表示第 i 个样本, 其具有 r 个属性, 即 $x_i=(x_{i1}, x_{i2}, \cdots, x_{ir})$. 将 X 中的样本构建为无权无向的全连接网络, 其对应的网络用 $G=(V, E)$ 表示, $V=\{v_1, v_2, \cdots, v_n\}$是网络中节点的集合, $E=\{e_1, e_2, \cdots, e_m\}$表示网络中边的集合, $e_m=\{(v_i, v_j)|v_i\neq v_j\wedge v_i, v_j\in V\}$. 经过网络构建, 原数据集 X 中的样本 x_i 对应网络 G 中的节点 v_i.

网络构建的目标是将空间中具有较近距离的样本之间建立连边关系, 且构建出的网络是一个连通图, 节

点之间的连边尽量稀疏. 本文采用 kNN 和ε-radius 组合方法进行网络构建. 组合方法可以确保所有样本点都出现在网络中, 且高密度区域的样本点具有较多的连边关系.

ε-radius 方法对样本 x_i 寻找半径为 eps 的范围内的其他样本 $NB_i=\{x_j|\text{dist}(x_i, x_j)<eps\}$, 如果$|NB_i|\geq k$, 则构建网络中的节点 v_i 与 NB_i 中所有样本对应节点的连边. 若$|NB_i|<k$, 则根据 kNN 方法进行边的构建.ε-radius 方法使得欧氏空间中高密度区域的节点在网络中具有稠密的连边关系, 连边越多, 该节点在网络中就越重要, 利于后续的社区发现.

kNN 方法对样本 x_i 寻找欧氏空间中距离最近的 k 个样本 $NN_i=\{x_j|$与 x_i 最近的 k 个节点$\}$, 并在对应的网络中构建节点 v_i 与 NN_i 中样本对应节点的连边. kNN 方法保证空间中处于边缘的点 (噪音点) 也能够与网络中其它节点建立连边关系, 使得数据集 X 中的所有样本都能出现在网络中.

2.2 标签预处理

LPA 算法步骤 1 中, 初始标签分布散乱, 标签传播的随机性较大, 容易导致标签误传播, 并增加算法的迭代次数. 一种可行的方法是对标签进行预处理, 使得相似度较高的节点具有相同的标签, 提升后续标签传播的稳定性, 缩减迭代次数.

用节点的共同邻居数衡量节点的相似度是一种常用的方法, 其定义如式 (3):

$$CN_{i,j}=\left|\Gamma_i\cap\Gamma_j\right| \tag{3}$$

其中, $v_j\in\Gamma_i$, 即只计算当前节点 v_i 和邻居节点的相似度.

定义 1. 标签初始化规则. 对于$\forall v_j\in\Gamma_i$, 计算 $CN_{i,j}$, 将 $CN_{i,j}$ 值最大的节点对应的标签作为 v_i 的标签. 式 (4) 求与 v_i 公共邻居数最大的节点编号 k. 当与 v_i 具有最大公共邻居数的节点不止一个时, 随机选择一个节点标签作为 v_i 的标签, v_i 的标签记为 lb_i. 标签初始化规则定义为式 (5):

$$k=\arg\max_{j\in\Gamma_i} CN_{i,j} \tag{4}$$

$$lb_i=\begin{cases} lb_k, & |k|=1 \\ lb_{rand(k)}, & |k|>1 \end{cases} \tag{5}$$

其中, $rand(k)$ 表示从集合 k 中随机选择一个.

2.3 融合节点影响力的标签传播

标签传播阶段, LPA 算法选择标签的方法是统计邻居节点的标签, 一个标签对应一组节点, 选择对应节点数最多的标签作为当前节点的标签. 有多个标签满足条件的, 就随机选择一个标签. 在节点连边较为稀疏的情况下, 尤其是当节点处于两个社区的连接路径上时, 以上方法容易造成标签误传播. 引入节点影响力, 在需要随机选择的情况下, 依据标签对应的节点 LR 值之和进行辅助选择, 进一步消除标签传播的随机性.

定义 2. 融合节点影响力的标签传播规则. 节点 v_i 的邻居节点中可能存在多个标签, 统计每个标签对应的节点数量, 某个标签对应的节点数量最多, 则选择该标签作为 v_i 的标签. 式 (6) 统计节点 v_i 邻居节点的标签归属情况, 并选出最大者:

$$maxk=\arg\max_k \sum_{j\in\Gamma_i^k}\delta_{ij} \tag{6}$$

当$|maxk|=1$ 时, $lb_i=lb_{maxk}$. 当$|maxk|>1$ 时, 进一步计算每个标签对应节点的影响力 (LR) 之和, 选择节点 LR 值之和最大的标签作为目标标签. 式 (7) 计算标签对应的节点影响力之和, 并选择 LR 值之和最大者 l 作为节点 v_i 的标签:

$$l=\arg\max_{k\in maxk}\sum_{j\in\Gamma_i^k} LR_j \tag{7}$$

其中, Γ_i^k 表示节点 v_i 的邻居节点中标签为 k 的节点集合. 因为此处讨论的节点 v_j 是 v_i 的邻居节点, 所以 $\delta_{ij}=1$.

2.4 社区优化合并

将向量表示的数据集进行网络化, 一般情况下要求构建的网络在确保全连通的前提下尽量稀疏, 网络中节点度通常不满足幂律分布. 用社区发现算法进行节点聚类, 在未指定社区数量的情况下, 得到的社区数通常会大于真实的簇的数量, 所以需要进行优化合并.

定义 3. 内度与外度. 假设 $C=\{c_1, c_2,\cdots, c_l\}$是 G 的一次社区划分结果, c_l 称为一个社区. 节点 $v_i\in c_l$ 的内度表示 v_i 与社区 c_l 内部节点的连边数量, 记为 $d_i^{\text{in}}(c_l)$. 外度表示 v_i 与社区 c_l 外部节点的连边数量, 记为 $d_i^{\text{out}}(c_l)$.

定义 4. 社区内聚度. 社区内聚度定义为社区 c 中的节点的内度之和与外度之和的比值, 比值越大, 表示内聚度越高, 社区质量越好. 当比值小于设定的阈值时, 该社区需要与相邻社区合并. 社区内聚度表示为 coh_c:

$$coh_c=\frac{\sum_{i\in c} d_i^{\text{in}}(c)}{\sum_{i\in c} d_i^{\text{out}}(c)} \tag{8}$$

定义 5. 社区优化合并规则. 当 $coh_c<\gamma$, 社区 c 需要与相邻社区进行合并, 选择 c 中最多外度所归属社区作为目标合并社区, 如式 (9):

$$t=\arg\max_{l}\sum_{i\in c}|d_i^{\text{out}}\in c_l|, c\neq c_l \tag{9}$$

当$|t|$=1 时, $lb_{i\in c}=t$; 当$|t|$>1 时, $lb_{i\in c}=rand(t)$.

2.5 算法伪代码

根据以上定义, 本文算法分为 4 个步骤: 首先对数据集进行网络化; 之后, 利用节点相似度对节点标签进行预处理, 以提高后续标签传播的稳定性; 在标签传播阶段, 用节点影响力辅助标签选择, 进一步降低标签传播的随机性; 最后, 通过对社区的内聚度进行判断, 对内聚度较小的社区进行合并优化, 以提高社区的质量.

算法 1. CreatGraph

输入: X,y, $maxk$, eps
输出: G

```
1  X=MinMaxScaler(X)
2  dist=kNN(X, maxk)
3  Foreach x_i∈X do
4      NB_i={x_j|dist(x_i, x_j)<eps}
5      if |NB_i|≥k then
6          G.addEdge(x_i, NB_i)
7      else
8          NN_i={x_j|与 x_i 最近的 k 个节点}
9          G.addEdge(x_i, NN_i)
10     end
11 end
12 return G
```

算法 1 用于将数据集转换为对应的网络, 其主要步骤如下:

1) 首先对数据进行最大最小值归一化, 即将各列特征值缩放到 [0,1] 之间, 以消除列之间特征值量纲不同引起的问题, 归一化的公式为:

$$x_{norm}=\frac{x-x_{\min}}{x_{\max}-x_{\min}} \tag{10}$$

2) 利用 kNN 算法求样本之间的距离, 并使用 k-d 树进行求解. 对于高维数据, 用 k-d 树可将时间复杂度降低为 O(Nlog$N$)(第 2 行);

3) 求与样本 x_i 的距离在半径 eps 范围内的样本点 (第 4 行);

4) 如果在半径距离内的样本数大于等于 k, 在图 G 中添加节点 v_i 与 NB_i 中所有样本对应的节点的边; 否则计算 NN_i, 并建立 v_i 与 NN_i 中节点的连边. 其中 *addEdge* 函数用于在节点对之间建立边, 因为构建的是无向图, 两个节点之间最多只建立一条边 (第 5–10 行);

5) 最后返回构建完成的网络 G (第 12 行).

算法 2. InitLabel

输入: $G=(V, E)$, γ
输出: LB

```
1  LB={lb_i=i|v_i∈V}
2  Foreach v_i ∈V do
3      k=arg max_{j∈Γ_i} CN_{i,j}
4      if |k|==1 then
5          lb_i=lb_k
6      else
7          lb_i=lb_{rand(k)}
8      end
9      update(LB, lb_i)
10 end
11 return LB
```

算法 2 根据相邻节点间的共同邻居数对节点进行标签初始化, 主要步骤如下:

1) 初始化节点标签为其对应的编号 (第 1 行);

2) 计算节点 v_i 与邻居节点的共同邻居数, k 中保留与 v_i 具有最多共同邻居数的节点标签 (第 3 行);

3) 根据定义 1 对标签进行预处理 (第 4–8 行);

4) 用新的标签更新标签集合 LB, 此处采用异步更新 (第 9 行).

算法 3. InfluLPA

输入: G, LB
输出: C

```
1  LR=LeaderRank(G)
2  finished=false
3  LBO=LB
4  While not finished do
5      LBN=Φ
6      Foreach v_i ∈V do
7          maxk=arg max_k Σ_{j∈Γ_i^k} δ_ij
8          if |maxk|=1 then
9              lb_i=lb_maxk
10         else
11             l=arg max_{k∈maxk} Σ_{j∈Γ_i^k} LR_j
12             lb_i=lb_l
13         end
14         LBN=LBN∪lb_i
15     end
16     if LBN==LBO then
17         finished=true
```

```
18      else
19          LBO = LBN
20      end
21 end
22 C=part(LBO)
23 return C
```

算法 3 的主要步骤如下:

1) 用 LeaderRank 算法计算节点的 *LR* 值, 用于后续的标签选择 (第 1 行);

2) 根据定义 2 进行一趟标签传播 (第 6–15 行);

3) 一趟标签传播后, 如果所有标签未发生变化, 迭代结束; 否则进行下一趟的标签传播 (第 16–20 行);

4) *part* 函数根据节点的标签将节点划分为不同的社区 (第 22 行).

算法 4. CombComm

输入: C, γ
输出: C'

```
1   C'=C
2   Foreach c ∈ C do
3       coh_c = (Σ_{i∈c} d_i^in(c)) / (Σ_{i∈c} d_i^out(c))
4       if coh_c<γ then
5           t=arg max_l Σ_{i∈c} |d_i^out ∈ c_l|, c≠c_l
6           if |t|==1 then
7               lb_{i∈c}=t
8           else
9               lb_{i∈c}=rand(t)
10          end
11          c_t=c_t∪c
12          C'=updata(C', c, c_t)
13      end
14 end
15 return C'
```

算法 4 根据社区内聚度进行社区优化合并, 其主要步骤如下:

1) 复制原始社区 C 到 C' (第 1 行);

2) 根据定义 4 计算当前社区的内聚度 (第 3 行);

3) 当内聚度小于阈值γ, 根据定义 5 计算合并的目标社区 t, 将 c 中节点的标签更改为 t (第 5–10 行);

4) 对 C'中的社区进行更新, 删除社区 c, 用更新后的社区 c_t 更新 C' (第 11–12 行).

2.6 时间复杂度分析

算法 1 中进行数据归一化的时间复杂度为 O(N); 利用基于 k-d 树的 kNN 算法求 N 个节点的最近邻节点和距离的时间复杂度为 O(Nlog$N$); 第 3–11 行构建图的时间复杂度为 O(N). 算法 1 的总体时间复杂度为 O(Nlog$N$).

算法 2 中初始化节点标签的时间复杂度为 O(N); 设节点的平均度为 k, 计算无向图中相邻节点的共同邻居数的时间复杂度为 O(k), 每个节点需要与 k/2 个邻居节点求交集, 则求 N 个节点与邻居节点相似度的时间复杂度为 O(Nk^2/2). 之后, 当前节点从邻居节点中选择一个标签的时间为 O(k), 为 N 个节点选择标签的时间复杂度为 O(kN). 算法 2 的总体时间复杂度为 O(Nk^2/2+Nk+k), 又因为 k=2M/N, 则总时间复杂度可简单表达为 O(kM).

算法 3 中, 计算节点的 *LR* 值的时间复杂度为 O(M+N), 第 2–21 行的主体是 LPA 算法, 其时间复杂度为 O(TM+N), T 为迭代次数. 第 11 行需要计算节点的 *LR* 值之和, 最坏情况下, 一次计算的时间复杂度为 O(k), 一般需要执行该计算的次数小于 0.1×N, 本文研究中的 k 值一般小于 10, 即最坏情况下该步骤的时间复杂度为 O(N); 最后, 将节点划分到社区所需的计算量为 O(N). 所以, 算法 3 总的时间复杂度为 O(TM+N).

算法 4 中, 计算一个节点的内度和外度所需时间为 O(k), 计算所有社区的内聚度需要计算所有节点的内度和外度, 其时间复杂度为 O(kN); 对于不满足内聚度的社区, 需要查询节点外度的归属社区, 所需查询的节点数小于 N, 则该步骤所需的计算机小于 O(kN); 第 12 行更新社区的计算量为 O(N). 算法 4 的总体时间复杂度为 O(kN), 即 O(M).

综上, 以上 4 个步骤的总体时间复杂度为 O(Nlog$N$+kM+TM+N+M), 实验结果表明, 迭代次数 T 一般小于 10, 节点度 k 也一般小于 10, 则时间复杂度可简化为 O(M+Nlog$N$).

3 实验及结果分析

为验证算法的有效性, 本文选取 Sklearn 工具包中的 K-means, DBSCAN 和 Spectral Clustering (SC) 3 个算法作为对比算法, 各算法在不同数据集上的参数设定以取得最大 *NMI* 值为准则进行设置. 实验数据为 15 次运行结果的平均值。

实验环境: Intel(R) Core(TM) i7-8650U CPU, 16 GB 内存, Windows 10 操作系统, 算法采用 Python 3.7 实现.

3.1 实验数据集

本文用 Sklearn 工具包中的 make_blobs 函数生成 4 个人工数据集, 用 make_circles 函数生成一个数据集, 并选择 UCI 上的 Iris, Wine, Letter-recognition(LR), WDBC 和 Glass 等 5 个数据集进行实验. 真实数据集的样本数量、特征数和簇数如表 1 所示, make_blobs 的参数值设定如表 2 所示, make_circles 的参数设定为 n_samples=160, noise=0.1, factor=0.5. 为方便后续描述, 将 make_circles 生成的数据集命名为 N5.

表 1 真实数据集

数据集	样本数	特征数	簇数
Iris	150	4	3
Wine	178	13	3
WDBC	569	30	2
Glass	214	9	6
LR	20 000	16	26

表 2 make_blobs 参数值

数据集	样本数	特征数	中心点	簇标准差
N1	100	2	2	1,3
N2	100	2	3	1,3,2
N3	160	2	4	1,3,2,2
N4	200	2	5	1,3,2,2,1

3.2 评价指标

对于已知样本标签的数据集, 可采用标准化互信息 (*NMI*) 和调整兰德系数 (*ARI*) 两个指标对聚类算法准确性进行评价.

NMI 定义为:

$$NMI(A,B)=\frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B}M_{ij}\log\left(\frac{M_{ij}N}{M_{i\cdot}M_{\cdot j}}\right)}{\sum_{i=1}^{C_A}M_{i\cdot}\log\left(\frac{M_{i\cdot}}{N}\right)+\sum_{j=1}^{C_B}M_{\cdot j}\log\left(\frac{M_{\cdot j}}{N}\right)} \tag{11}$$

其中, A 是样本真实标签, B 是算法聚类后的样本标签, N 是样本数. C_A 是真实簇数量, C_B 是经算法聚类的簇数量. M 是混淆矩阵, $M_{i\cdot}$ 是矩阵 M 中第 i 行元素之和, 表示 A 中第 i 个簇的样本数. 相应的, $M_{\cdot j}$ 表示 B 中第 j 个簇的样本数, 即矩阵 M 中第 j 列元素之和. M_{ij} 表示 A 中第 i 个簇的样本属于 B 中第 j 个簇的样本数. *NMI* 的值域为 [0,1], 值越大则表示算法的聚类效果越好, 当 $NMI(A,B)=1$ 时, 表示 A 和 B 的结构完全相同.

ARI 定义为:

$$ARI=\frac{\sum_{ij}\binom{n_{ij}}{2}-\left[\sum_{i}\binom{n_{i\cdot}}{2}\sum_{j}\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i}\binom{n_{i\cdot}}{2}+\sum_{j}\binom{n_{\cdot j}}{2}\right]-\left[\sum_{i}\binom{n_{i\cdot}}{2}\sum_{j}\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}} \tag{12}$$

其中, n 是混淆矩阵, n_{ij} 表示混淆矩阵中第 i 行第 j 列元素, $n_{i\cdot}$ 是混淆矩阵中第 i 行元素之和, $n_{\cdot j}$ 是混淆矩阵中第 j 列元素之和, $\binom{n}{2}$ 表示从 n 个节点中取 2 个节点的组合数. *ARI* 的取值范围为 [−1,1], 值越大, 说明社区划分结果与真实社区越吻合.

3.3 聚类精度实验

在 Iris 等 5 个真实网络上的实验结果如图 2 所示, 图 2(a) 是算法得到的 *NMI* 值, 图 2(b) 是算法得到的 *ARI* 值. 对于 *NMI* 指标, 本文算法在 Iris, WDBC, Glass 和 LR 等 4 个数据集上获得最优值. 在 Wine 数据集上, Spectral Clustering 算法取得最优值, 本文算法得到的结果优于 K-means 和 DBSCAN. 在 *ARI* 指标上得到的实验结果与在 *NMI* 上得到的结果类似.

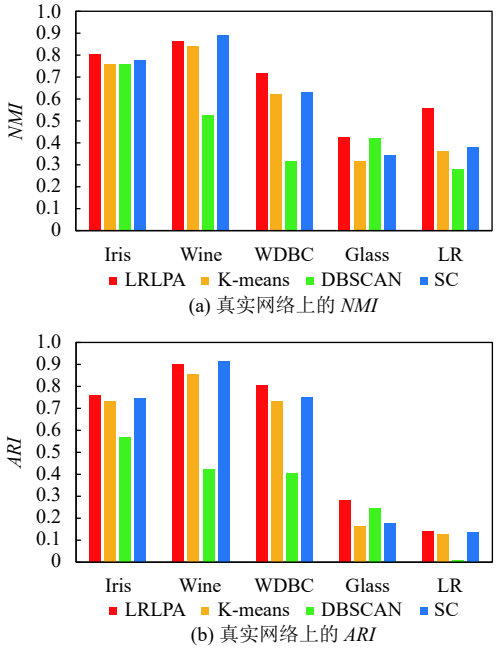
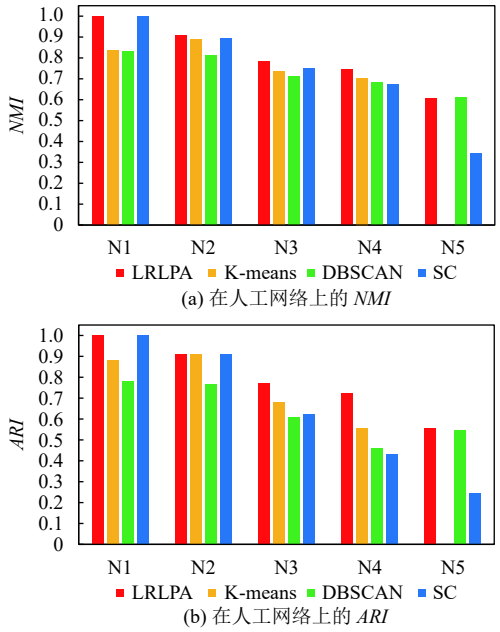


图 2 不同算法在真实网络上的精度实验

在人工数据集上, 本文算法也取得了较好的结果. 图 3(a) 中, 在 N1 数据集上, 本文算法和 Spectral

Clustering 都能够完全正确的对数据进行划分, 得到的 *NMI* 和 *ARI* 值都是 1. 在 N2 数据集上, 本文算法得到的结果与 K-means 和 Spectral 较为接近. 在 N3 和 N4 数据集上, 本文算法得到的结果明显优于对比算法. 图 3(b) 中的 *ARI* 实验结果与 *NMI* 的结果类似.



(a) 在人工网络上的 *NMI*

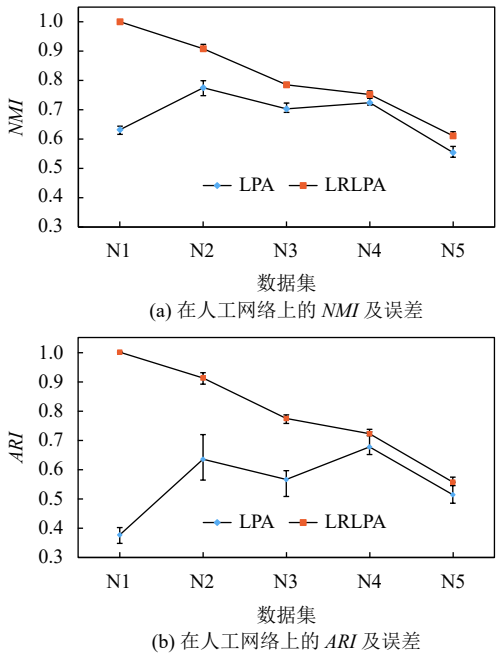
(b) 在人工网络上的 *ARI*

图 3 不同算法在人工网络上的精度实验

在前 4 个数据集中, 都存在标准差为 3 的簇, 使得数据集中存在较多的噪音点, DBSCAN 算法依赖于于样本密度, 对噪音点的识别能力较弱, 所以在 4 个数据集上的准确性都比较差. 本文算法在网络构建过程同时考虑了密度和 k 个最近邻样本, 保证了距离簇中心较远的样本也能与该簇中节点保持较多的连边, 有利于后续的社区发现. N5 数据集是两个同心圆, 两个圆之间有部分节点重叠, Spectral Clustering 算法无法区分簇之间的界限, 划分出的结果与真实情况差别较大, K-means 算法对这种特殊类型的图形无法识别, 得到的 *NMI* 和 *ARI* 都为 0, 而 LRLPA 和 DBSCAN 基于密度, 能够在密度较小的区间进行划分, 所以两者得到的结果比较接近.

3.4 算法稳定性实验

LRLPA 和 LPA 两种算法在人工数据集上的实验结果如图 4 所示. 无论是在 *NMI* 还是 *ARI* 指标上, 本文提出的 LRLPA 都比原始 LPA 具有更高的精确度, 且具有更小范围的误差. 对数据集 X 进行网络化后, 一般会比较稀疏, LPA 识别出的社区数较多, 使得准确度指标不高. 而 LRLPA 最后一步根据内聚度进行优化合并, 使得社区数与真实簇数相同或接近, 提高了算法的精确度. 同时, 标签预处理与融入节点影响力的标签传播对降低算法随机性有较为明显的效果, 在 4 个数据集上的波动范围都小于 2%.



(a) 在人工网络上的 *NMI* 及误差

(b) 在人工网络上的 *ARI* 及误差

图 4 LRLPA 和 LPA 的精度即稳定性实验

4 结论与展望

本文提出了一种基于网络社区发现的低随机性标签传播聚类算法 LRLPA. 在 5 个真实数据集和 5 个人工数据集上进行了实验, 与其他算法相比, LRLPA 算法在大部分数据集上都具有最大的 *NMI* 和 *ARI* 值. 与 LPA 的对比中, 本文算法的准确率和稳定性都高于 LPA.

下一步研究中, 一个改进的方向是通过动态网络构建, 将 LRLPA 算法应用于增量式聚类. 另一个值得注意的方向是改进标签传播方式, 将标签选择过程并行化, 提高算法效率, 以适应大规模数据集的应用.

参考文献

1 雷小锋, 陈皎, 毛善君, 等. 基于随机 kNN 图的批量边删除聚类算法. 软件学报, 2018, 29(12): 3764−3785. [doi: 10.13328/j.cnki.jos.005327]

2 Kuwil FH, Atila Ü, Abu-Issa R, *et al*. A novel data clustering

algorithm based on gravity center methodology. Expert Systems with Applications, 2020, 156: 113435.
3 Ester M, Kriegel HP, Sander J, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland, OR, USA. 1996. 226−231.
4 郭艳婕, 杨明, 侯宇超, 等. 基于相似性度量的改进 DBSCAN 算法. 数学的实践与认识, 2020, 50(6): 164–170.
5 Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases. Proceedings of 1998 ACM SIGMOD International Conference on Management of Data. Seattle, WA, USA. 1998. 73−84. [doi: 10.1145/276304.276312]
6 Ng AY, Jordan MI, Weiss Y. On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems. Vancouver, BC, Canada. 2001. 849−856.
7 胡卓娅, 翁健. 基于人工蜂群算法的自适应谱聚类算法. 重庆理工大学学报 (自然科学), 2020, 34(3): 137–144.
8 Xie WB, Lee YL, Wang C, *et al*. Hierarchical clustering supported by reciprocal nearest neighbors. Information Sciences, 2020, 527: 279–292.
9 Silva TC, Zhao L. High-level pattern-based classification via tourist walks in networks. Information Sciences, 2015, 294: 109–126.
10 Girvan M, Newman MEJ. Community structure in social and biological networks. Proceedings of the National Academy of Sciences of the United States of America, 2002, 99(12): 7821–7826.
11 Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 2007, 76(3): 036106.
12 Zhu X, Ghahramani Z. Learning from labeled and unlabeled data with label propagation. Pittsburgh, PA, USA: Carnegie Mellon University, 2002.
13 Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 1970, 49(2): 291–307.
14 Fiedler M. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 1973, 23(2): 298–305.
15 Newman MEJ. Spectral methods for community detection and graph partitioning. Physical Review E, 2013, 88(4): 042822.
16 Lin CC, Kang JR, Chen JY. An integer programming approach and visual analysis for detecting hierarchical community structures in social networks. Information Sciences, 2015, 299: 296–311.
17 Xu XW, Yuruk N, Feng ZD, *et al*. SCAN: A structural clustering algorithm for networks. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Jose, CA, USA. 2007. 824–833.
18 Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Physical Review E, 2004, 69(2): 026113.
19 Blondel VD, Guillaume JL, Lambiotte R, *et al*. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10): P10008.
20 Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences of the United States of America, 2008, 105(4): 1118–1123.
21 Sun ZJ, Wang B, Sheng JF, *et al*. Community detection based on information dynamics. Neurocomputing, 2019, 359: 341–352. [doi: 10.1016/j.neucom.2019.06.020]
22 Lü LY, Zhang YC, Yeung CH, *et al*. Leaders in social networks, the delicious case. PLoS One, 2011, 6(6): e21202.
23 Li Q, Zhou T, Lü LY, *et al*. Identifying influential spreaders by weighted LeaderRank. Physica A: Statistical Mechanics and Its Applications, 2014, 404: 47–55.
24 Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 1998, 30(1–7): 107–117.
25 刘世超, 朱福喜, 甘琳. 基于标签传播概率的重叠社区发现算法. 计算机学报, 2016, 39(4): 716–729.